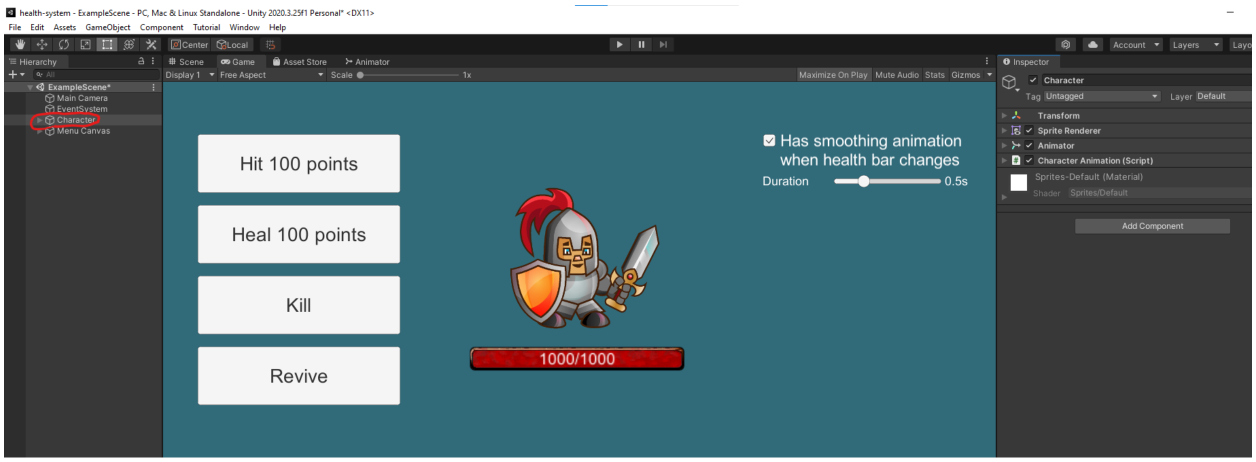
Health System for Dummies Guide

Index

How to add the Health System For Dummies to a GameObject

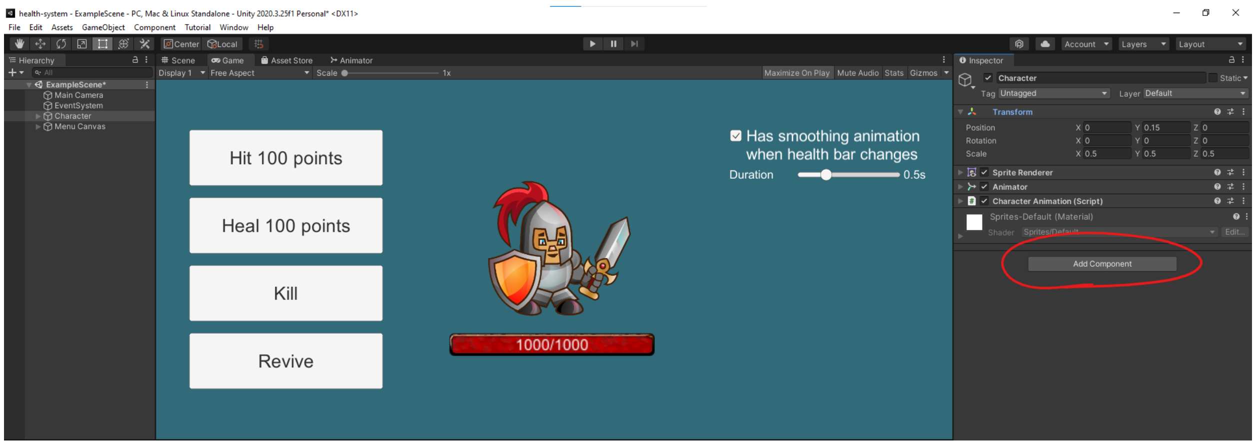
1. In your Scene's hierarchy, select the GameObject you'd like to have health, for instance, here I selected the GameObject `Character`

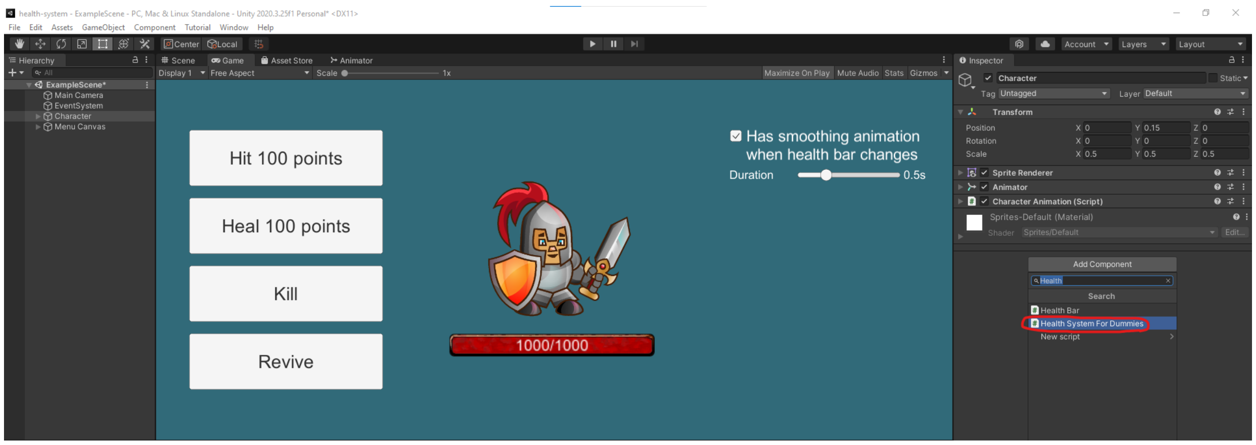


☑ Has smoothing animation when health bar changes
Duration 0.5s

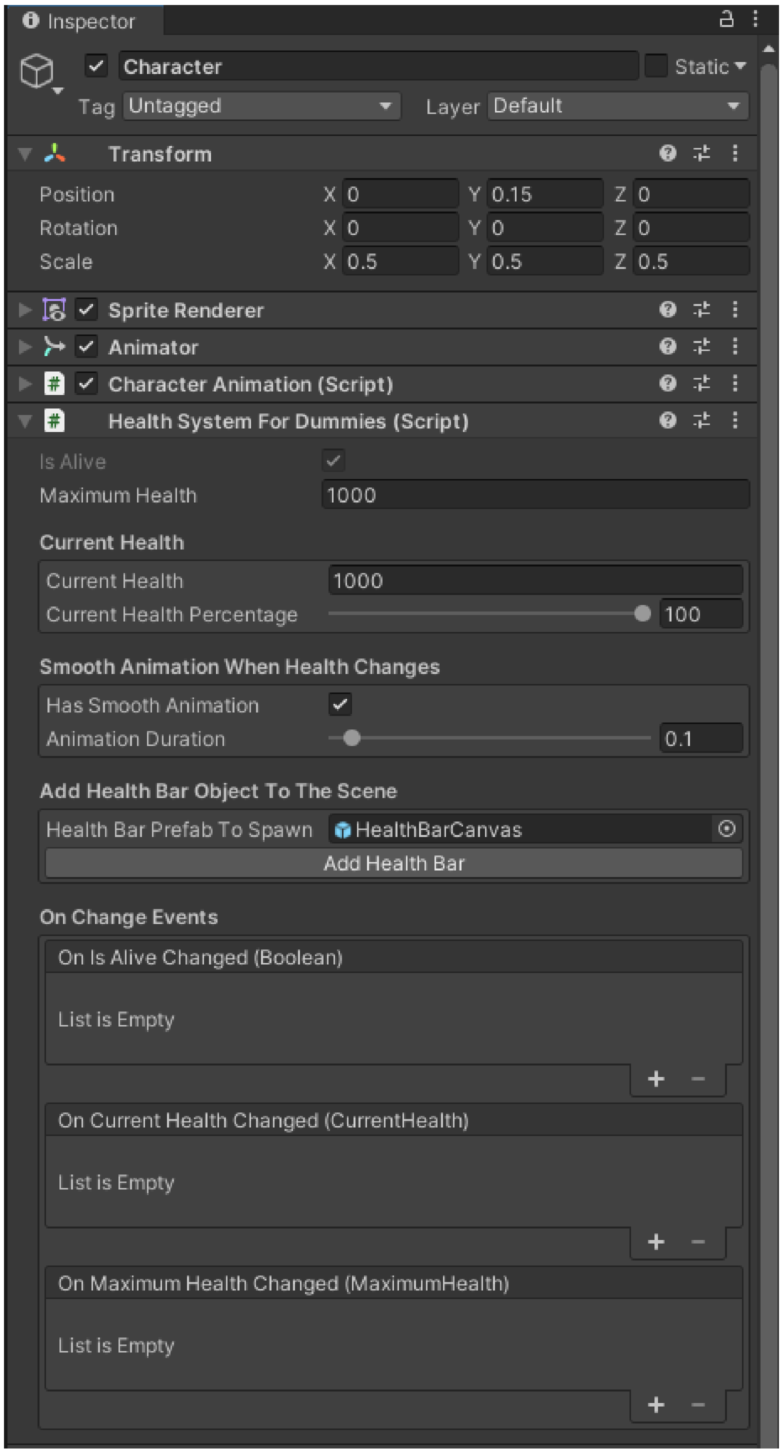
Character
Tag Untagged Layer Default
Shader Sprites/Default

2. In the Inspector window, click on `Add Component`, search for a component named **Health System For Dummies** and click it





3. Now that you've added it, you'll see the **Healt hSystem For Dummies script** in the Inspector window.
   Because it has loads of properties, I'm going to explain them thoroughly down below.



Character ✓ — Static

Tag: Untagged — Layer: Default

Transform

	X	Y	Z
Position	0	0.15	0
Rotation	0	0	0
Scale	0.5	0.5	0.5

✓ Sprite Renderer

✓ Animator

✓ Character Animation (Script)

Health System For Dummies (Script)

Is Alive ✓

Maximum Health: 1000

Current Health

- Current Health: 1000
- Current Health Percentage: 100

Smooth Animation When Health Changes

- Has Smooth Animation ✓
- Animation Duration: 0.1

Add Health Bar Object To The Scene

- Health Bar Prefab To Spawn: HealthBarCanvas
- Add Health Bar

On Change Events

- On Is Alive Changed (Boolean): List is Empty
- On Current Health Changed (CurrentHealth): List is Empty
- On Maximum Health Changed (MaximumHealth): List is Empty

Health System Properties

Is Alive → This field will tell you if the GameObject still has any health left

Maximum Health → This field determines the GameObject's maximum health. It can be increased/decreased as you wish

Current Health

Current Health → This field determines the GameObject's flat current health. It can be increased/decreased as you wish

Current Health Percentage → This field determines the GameObject's current health percentage. It can be increased/decreased as you wish

Smooth Animation When Health Changes

Has Smooth Animation → If checked, whenever the GameObject's current health is increased/decreased, the health bar will take as long as the **Animation Duration** time to reach it's actual value. This will make it seem like a more organic health change to the user, but can be disabled if you want to

Animation Duration → How much time you want the smooth animation described in the field above to take.

Add Health Bar Object To The Scene

Health Bar Prefab To Spawn → Field that already comes with a default **Health Bar GameObject** that will be spawned when you click the **Add Health Bar** button. You can change it to another prefab of yours if you wish.

When you click the **Add Health Bar** button, a Health Bar will be created in your Scene in **World Space,** so it will be a visible object that will follow the GameObject it is a child of.

On Change Events

This section is going to be explained later on in this guide, because it is a bit advanced.

How to interact with the Health System

These are all the methods you can interact with from the HealthSystemForDummies:

AddToMaximumHealth(float value): Increases or decreases the GameObject’s maximum health

AddToCurrentHealth(float value): Damages the GameObject if the `value` argument is negative, and heals the GameObject if the `value` argument is positive

ReviveWithMaximumHealth(): GameObject gets revived and will have maximum health

ReviveWithCustomHealth(float healthWhenRevived): GameObject gets revived and will have as much health as is set by the argument `healthWhenRevived`

ReviveWithCustomHealthPercentage(float healthPercentageWhenRevived): GameObject gets revived and will have as much health percentage as is set by the argument `healthWhenRevived` (the value here should be from 0 to 100)

Kill(): Kills the GameObject and turn its current health to 0.

Examples on how to use them

1. Say you wanted to revive your character that has the Health System component, every time you press the `R` key on the keyboard.

   You could do something like that:

```
class Character : MonoBehaviour {
    HealthSystemForDummies healthSystem;

    void Start() {
        healthSystem = GetComponent<HealthSystemForDummies>();
    }

    void Update() {
        if (Input.GetKeyUp(KeyCode.R)) {
            healthSystem.ReviveWithMaximumHealth();
        }
    }
}
```

2a. Say you wanted to damage an enemy for -100 whenever a bullet hits it.

You could do something like that:

```
class Bullet : MonoBehaviour {
    void OnCollisionEnter(Collision collision) {
        if (collision.transform.CompareTag("Enemy")) {
            HealthSystemForDummies healthSystem = collision.gameObject.GetComponent<HealthSystemForDummies>();

            // Damage enemy for -100 units
            healthSystem.AddToCurrentHealth(-100);
        }
    }
}
```

2b. Or kill the enemy whenever a bullet hits it:

```
class Bullet : MonoBehaviour {
    void OnCollisionEnter(Collision collision) {
        if (collision.transform.CompareTag("Enemy")) {
            HealthSystemForDummies healthSystem = collision.gameObject.GetComponent<HealthSystemForDummies>();

            // Kills the enemy
            healthSystem.Kill();
        }
    }
}
```

(These scripts consider that there is a GameObject tagged as `Enemy` that has the `HealthSystemForDummies` component)

On Change Events explanation

This section can be used to "listen" to when an event has happened. So for instance if the current health goes from 1000 to 900, a `On Current Health Changed` event will happen. These events can be "listened" to and can call a method of your choice. It is a bit confusing but I think it will become clearer as you follow through this guide.

There are two main ways to listen to an event. The easier way (Unity Events in Inspector) and a more advanced one (listening to events via C# code). Let's go through each of them:

1. Unity Events in Inspector

Say you want to know if your GameObject is alive or not. You can do it by listening to the **On Is Alive Changed (Boolean)** event. To do that, in a script you have to create a method that has a boolean as a parameter and then add it in the event listener list. For example, in a script **ExampleScript.cs,** you could create a method like:

```
void PlayAnimation(bool isAlive)
{
    bool characterIsDead = !isAlive;

    if (characterIsDead)
    {
        PlayDeadAnimation();
    }
}
```

Here we created a method that has a boolean parameter (that's the `bool isAlive` part of the method) because the event in the Inspector asked us for a Boolean method. I know that because of the type described in parenthesis: On Alive Changed → **(Boolean)** ←.

Now all we have to do is:

1. In the Inspector, inside the Health System script, click the + icon in the event we want to listen to
2. Drag the GameObject that has the ExampleScript Component from the Hierarchy window to the GameObject field (the field that has the text **None (Object)**)
3. Click on the **No Function** field, then in the dropdown find the ExampleScript and then the PlayAnimation (bool) method

That's it 🚀
Now every time `isAlive` is changed, the `PlayAnimation` method will be called and if the GameObject is not alive anymore, the **Dead** animation will be called

2. **Listening to events via C# code**

Say you want to know if your GameObject is alive or not. You can do it by listening to the **On Is Alive Changed (Boolean)** event. To do that, in a script you have to create a method that has a boolean as a parameter and then add it in the event listener list. For example, in a script **ExampleScript.cs,** you could create a method like:

```
void PlayAnimation(bool isAlive)
{
    bool characterIsDead = !isAlive;

    if (characterIsDead)
    {
        PlayDeadAnimation();
    }
}
```

Now to listen to OnIsAliveChanged, make sure you have a reference to HealthSystemForDummies, like:

```
HealthSystemForDummies healthSystem;

class ExampleScript {
    void Start() {
        healthSystem = GetComponent<HealthSystemForDummies>();
    }
}

void PlayAnimation(bool isAlive)
{
    bool characterIsDead = !isAlive;

    if (characterIsDead)
    {
        PlayDeadAnimation();
    }
}
```

Then, to listen to the Unity Event, add a listener like shown in bold:

```
HealthSystemForDummies healthSystem;

class ExampleScript : MonoBehaviour {
    void Start() {
        healthSystem = GetComponent<HealthSystem>();
        healthSystem.OnChangedIsAlive.AddListener(PlayAnimation);
    }
}

void PlayAnimation(bool isAlive)
{
    bool characterIsDead = !isAlive;

    if (characterIsDead)
    {
        PlayDeadAnimation();
    }
}
```

That's it 🚀
Now every time `isAlive` is changed, the `PlayAnimation` method will be called and if the GameObject is not alive anymore, the **Dead** animation will be called